PILC:
Performance Enhancing Proxies (PEPs)

46th IETF

draft-ietf-pilc-pep-xx.txt

John Border, Markku Kojo,
Jim Griner, Gabriel Montenegro

Presentation Outline

- Purpose of the draft
- Overview of the draft
  - Types of PEPs
  - PEP Mechanisms
  - PEP Examples
  - PEP Implications
- Status and next steps

Purpose of the Draft

- The purpose of this draft is different from the purpose of the other PILC drafts. This draft is intended to document:
  - What PEPs are and how they are implemented;
  - What motivates their development and use for particular links;
  - What the implications are of using them, especially with respect to the end to end argument.
- The draft is not trying to define any sort of standards related to PEPs or their use. It is just trying to capture “current art”.
- The draft is not trying to make recommendations for or against the use of PEPs except by means of the implications of their use.

Types of PEPs

- PEPs can be classified by the layer at which they operate:
  - Transport layer (e.g. TCP) versus application layer
- PEPs can be classified by the implementation distribution:
  - A PEP implementation can be integrated into a single node.
    - Example: A PEP which provides impedance matching between wired and wireless links.
  - A PEP implementation can be distributed between several nodes.
    - Example: Two PEPs located at each end of a satellite link to improve performance over the link.

Types of PEPs (Cont.)

- PEPs can be classified by their treatment of connections:
  - A PEP implementation can assist connections without “interfering” with their end to end flow
    - Example: A PEP implementation which simply spaces TCP ACKs to reduce traffic burstiness.
  - A PEP implementation can split an end to end connection into multiple connections
    - Example: Two PEPs at the ends of a satellite link which terminate TCP connections at each PEP and use a third connection between the two PEPs.

Types of PEPs (Cont.)

- PEPs can be classified by their degree of transparency:
  - A PEP implementation may require changes to neither, one or both of the end systems of a connection.
  - Transparency is an issue at multiple levels: the network layer, the transport layer, the application layer and the user.
- PEPs can be classified by their their degree of interference with the end to end semantics of a connection.
  - Related to, but not the same as, transparency.
  - Not in the draft yet.

PEP Mechanisms

- A PEP may use one or more of the following mechanisms to try to improve performance:
  - ACK spacing
  - ACK regeneration (not in the draft yet)
  - Local acknowledgements
  - Local retransmissions
  - Tunnels to control routing of packets
  - Header compression
  - Payload compression
  - Priority based multiplexing
  - Others?

PEP Examples

- The draft includes several examples of environments where PEPs are used:
  - Satellite VSAT networks
  - [Mobile] Wireless WAN (W-WAN) networks
  - Wireless LAN (W-LAN) networks
  - Wireless Application Protocol (WAP) networks (not in the draft yet)
- The examples are provided to try to give substance to the various PEP types and mechanisms, making them easier to understand.
- Many references to PEP implementations are included to provide additional detail. Every type and mechanism is backed up by at least one reference (hopefully).

PEP Implications

- Many of the implications of using PEPs relate to the end to end argument.
  - The use of a PEP should never be transparent to the user.
    - When “user” is defined to include the network administrator, most existing PEP implementations are non-transparent to the user.
      - PEPs are primarily used today in an intranet or “last hop” Internet context.
  - The implementation of a PEP should allow user control over which connections are “PEPed” and which connections are not “PEPed”.
    - Some, but definitely not all, existing PEP implementations provide this sort of user control.

PEP Implications (Cont.)

- Re the end to end security argument:
  - Since PEPs need to see inside IP packets and, in some implementations, generate IP packets on behalf of an end system, PEPs cannot be used with end to end IPsec.
    - Using end to end IPsec prevents the use of PEPs.
    - The desire to use PEPs keeps a user from using end to end IPsec.
  - Tunneled IPsec could be used with PEPs as the tunnel end points.
    - Requires the PEPs to be trusted by the user.
    - In general, security mechanisms at or above the transport layer (e.g. TLS or SSL) can be used with PEPs.
  - Multi-layer IPsec? [draft-zhang-ipsec-mlipsec-00.txt]

PEP Implications (Cont.)

- Re the end to end fate sharing argument:
  - Most PEP implementations keep state.
    - A failure of a PEP implementation which only keeps “soft” state may support failover to alternate paths.
    - A failure of a PEP implementation which keeps “hard” state (e.g. state required to support split connections) will generally cause a connection to fail even though an alternate end to end path exists for the connection.
      - Note that “hard” state is not strictly related to the use of split connections.
      - Sometimes coincidentally and sometimes by design, PEPs are often positioned where no alternate path exists.

PEP Implications (Cont.)

- Re the end to end reliability argument:
  - A PEP implementation may affect the end to end reliability of a connection, especially if the PEP interferes with application layer acknowledgements.
    - Applications should not rely on lower level (e.g. TCP) acknowledgements to guarantee end to end delivery.
    - TCP PEPs generally do not interfere with application layer acknowledgements.
- Re the end to end failure diagnostics argument:
  - Using a PEP potentially interferes with the use of end to end failure diagnostics tools.

PEP Implications (Cont.)

- Other implications:
  - Using PEPs can place constraints on the routing topology:
    - Suboptimal routing might be required to force traffic to go through a PEP;
    - Tunnels might be required to force traffic to go through a PEP, especially in an asymmetric routing environment.
  - Using PEPs with mobile hosts might require PEP state to be handed off as the hosts move.
  - Others?

Status

- The -00 draft was released just prior to the Oslo IETF meeting.
  - We received some good input from several people.
- The -01 draft was submitted prior to this meeting but just missed the submission deadline.
  - The draft will be released right after the D.C. meeting.
  - The draft incorporates some, but not all, of the comments received so far.
- The plan is to release a -02 draft as quickly as possible after -01 draft, incorporating all of the rest of the comments.
- We need help filling in some of the sections of the draft.

Changes in the -01 Draft

- Split the “types of PEPs” and “PEP mechanisms” section into separate sections for clarity.
- Tried to clarify various sections:
  - End to end related issues;
  - Terminology:
    - Tried to make layer versus protocol distinctions more accurate;
    - Tried to make TCP versus application distinctions more accurate;
    - Changed uses of “proxy” to “PEP” to eliminate ambiguity (with respect to other uses of the term “proxy”).
- Added some new mechanisms such as prioritizing access to resources.

Soliciting Input

- We are current soliciting input for:
  - Terminology refinement;
  - Additional PEP types and/or mechanisms which should be included in the draft. Some ideas already suggested which need flushing out include: ACK regeneration, partial ACK mechanisms, priority based multiplexing and protocol booster mechanisms.
  - Additional example environments where PEPs are used.
    - However, we are not trying to describe every PEP implementation in existence. So, any proposed additions should illustrate types or mechanisms of PEPs other than those illustrated by the existing examples.
    - Need someone to flush out the section on WAP.

Soliciting Input (Cont.)

- Most importantly, we are current soliciting input on additional implications of using PEPs. Some ideas already suggested which need to be flushed out (or need additional flushing out) include:
  - Scalability
  - Failure diagnostics
  - Multi-homing environments
  - QoS transparency